PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-166931
(43)Date of publication of application : 25.06.1996

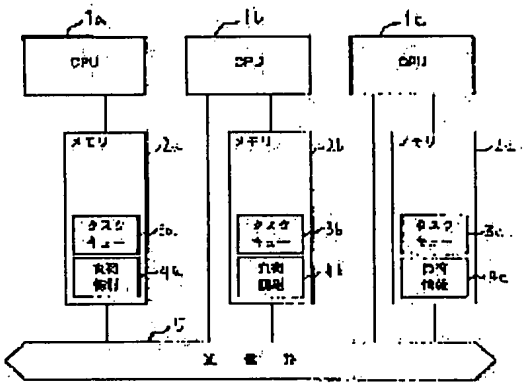
(51)Int.Cl. G06F 15/16

(21)Application number : 06-308906
(22)Date of filing : 13.12.1994

(71)Applicant : MITSUBISHI ELECTRIC CORP
(72)Inventor : SATO HIROAKI
AOYAMA KAZUHIRO

(54) LOAD DISTRIBUTION METHOD FOR PARALLEL COMPUTER

(57)Abstract:
PURPOSE: To reduce the communication load of a channel by transferring a task to CPU whose load is small in a parallel computer and executing it when the task cannot be processed in self CPU at the time of generating the task.
CONSTITUTION: CPU1a generates the task, reads the load situation of self CPU from load information 4a and judges whether the load situation is less than a permitted value or not. When the load situation is less than the permitted value, it is cconsidered that the generated task can be processed in self CPU1a, the task is registered in the task queue 3a of self CPU1a and a task generation processing is terminated. When the load situation is larger than the permitted value, it is considered that the generated task cannot be processed in self CPU1a, the load situation of other CPU1b and 1c in the parallel computer is read from load information 4b and 4c and CPU whose load is the smallest is selected from the load situation which is read.



LEGAL STATUS

[Date of request for examination] 06.06.2000
[Date of sending the examiner's decision of rejection] 06.08.2002
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開平8－166931
(43)公開日　平成８年(1996)６月25日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　15/16	３８０　Ｚ			

審査請求　未請求　請求項の数６　ＯＬ　（全 16 頁）

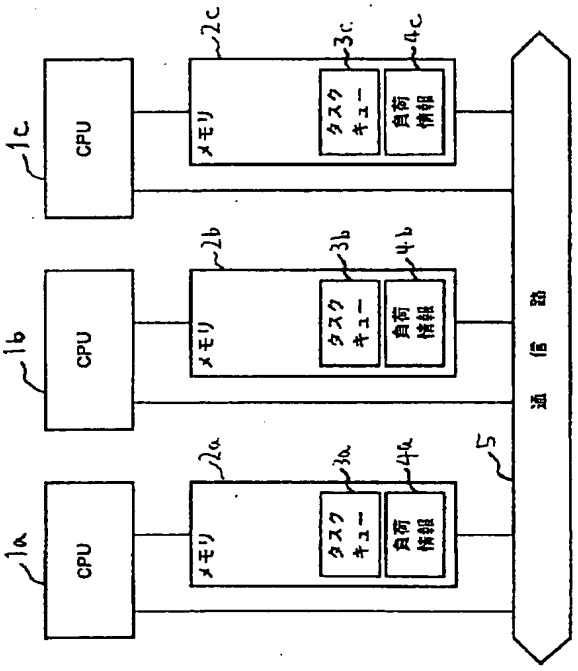
(21)出願番号　特願平6－308906

(22)出願日　平成６年(1994)12月13日

(71)出願人　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目２番３号
(72)発明者　佐藤　裕昭
鎌倉市上屋町325番地　三菱電機株式会社
鎌倉製作所内
(72)発明者　青山　和弘
鎌倉市上屋町325番地　三菱電機株式会社
鎌倉製作所内
(74)代理人　弁理士　高田　守　（外４名）

(54)【発明の名称】　並列計算機の負荷分散方法

(57)【要約】
【目的】　通信路の通信負荷が小さく、負荷分散のオーバーヘッドが少なく、ＣＰＵ構成の情報を局所化した、スケーラビリティのある並列計算機の負荷分散方法を得る。
【構成】　タスクキュー３ａ、３ｂ、３ｃをＣＰＵ１ａ、１ｂ、１ｃ毎に持たせ、タスクの生成時に自分のＣＰＵで処理できない場合にのみ、他のＣＰＵのタスクキューにタスクを登録し実行させる。



1

【特許請求の範囲】

【請求項１】　複数のＣＰＵを持つ並列計算機上で、各ＣＰＵの処理負荷がオーバーフローしないように処理負荷を分散する負荷分散方法において、各ＣＰＵが自分のＣＰＵの負荷状況を観測し、あるＣＰＵがタスクを生成する際に、自分のＣＰＵの負荷が大きく生成したタスクを処理できない場合、並列計算機内のＣＰＵの負荷状況を調べ、負荷の小さいＣＰＵにタスクを転送し実行させることを特徴とする並列計算機の負荷分散方法。

【請求項２】　複数のＣＰＵを持つ並列計算機上で、各ＣＰＵの処理負荷がオーバーフローしないように処理負荷を分散する負荷分散方法において、各ＣＰＵが自分のＣＰＵの負荷状況を観測し、あるＣＰＵがタスクを生成する際に、自分のＣＰＵの負荷が大きく生成したタスクを処理できない場合、ＣＰＵ間距離情報データベースからＣＰＵ間距離を取得し、並列計算機内のＣＰＵの負荷状況を調べ、負荷が小さくＣＰＵ間距離の短いＣＰＵにタスクを転送し実行させることを特徴とする並列計算機の負荷分散方法。

【請求項３】　複数のＣＰＵを持つ並列計算機上で、各ＣＰＵの処理負荷がオーバーフローしないように処理負荷を分散する負荷分散方法において、あらかじめ何も処理を割り当てておかない予備ＣＰＵを備え、各ＣＰＵが自分のＣＰＵの負荷状況を観測し、あるＣＰＵがタスクを生成する際に、自分のＣＰＵの負荷が大きく生成したタスクを処理できない場合、予備ＣＰＵにタスクを転送し実行させることを特徴とする並列計算機の負荷分散方法。

【請求項４】　複数のＣＰＵを持つ並列計算機上で、各ＣＰＵの処理負荷がオーバーフローしないように処理負荷を分散する負荷分散方法において、タスク配分ＣＰＵを備え、各ＣＰＵが自分のＣＰＵの負荷状況を観測し、あるＣＰＵがタスクを生成する際に、自分のＣＰＵの負荷が大きく生成したタスクを処理できない場合、タスク配分ＣＰＵにタスクを転送し、タスク配分ＣＰＵが並列計算機内のＣＰＵの負荷状況を調べ、負荷の小さいＣＰＵにタスクを転送し実行させることを特徴とする並列計算機の負荷分散方法。

【請求項５】　複数のＣＰＵを持つ並列計算機上で、各ＣＰＵの処理負荷がオーバーフローしないように処理負荷を分散する負荷分散方法において、タスク配分ＣＰＵを備え、各ＣＰＵが自分のＣＰＵの負荷状況を観測し、あるＣＰＵがタスクを生成する際に、自分のＣＰＵの負荷が大きく生成したタスクを処理できない場合、タスク配分ＣＰＵにタスクを転送し、タスク配分ＣＰＵがＣＰＵ間距離情報データベースからＣＰＵ間距離を取得し、並列計算機内のＣＰＵの負荷状況を調べ、負荷が小さくＣＰＵ間距離の短いＣＰＵにタスクを転送し実行させることを特徴とする並列計算機の負荷分散方法。

【請求項６】　複数のＣＰＵを持つ並列計算機上で、各

2

ＣＰＵの処理負荷がオーバーフローしないように処理負荷を分散する負荷分散方法において、タスク配分ＣＰＵと、あらかじめ何も処理を割り当てておかない予備ＣＰＵを備え、各ＣＰＵが自分のＣＰＵの負荷状況を観測し、あるＣＰＵがタスクを生成する際に、自分のＣＰＵの負荷が大きく生成したタスクを処理できない場合、タスク配分ＣＰＵにタスクを転送し、タスク配分ＣＰＵが予備ＣＰＵにタスクを転送し実行させることを特徴とする並列計算機の負荷分散方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】この発明は、並列計算機の負荷分散方法に関するものである。

【０００２】

【従来の技術】図１３は、従来の並列計算機の負荷分散方法を示すブロック図であり、図において、１ａ、１ｂ、１ｃはＣＰＵ、２ａ、２ｂ、２ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃに接続されているメモリ、３はＣＰＵ１ａ、１ｂ、１ｃが実行するタスクを保持しておくタスクキュー、５はメモリ２ａ、２ｂ、２ｃをＣＰＵ１ａ、１ｂ、１ｃで共有するための通信路である。

【０００３】次に動作について図１４（ａ）、（ｂ）を参照しながら説明する。図１４（ａ）、（ｂ）は図１３のＣＰＵ１ａ、１ｂ、１ｃで動作する並列計算機の負荷分散方法を示すフローチャートであり、図１４（ａ）はタスク生成処理を示し、図１４（ｂ）はタスク割当処理を示す。

【０００４】並列計算機の負荷分散方法を行うために、ＣＰＵ１ａ、１ｂ、１ｃは、タスクを生成する際に、Ｓ４５ステップでタスクを生成した後、Ｓ４６ステップでタスクをタスクキュー３に登録する。次に、ＣＰＵ１ａ、１ｂ、１ｃは、自分のＣＰＵで実行しているタスクが資源待ち状態などでブロッキングされた場合、Ｓ４７ステップで実行中のタスクをタスクキュー３に退避し、Ｓ４８ステップでタスクキュー３から別のタスクを取りだし、自分のＣＰＵに割り当てて実行する。このように、１つのタスクキューでタスクを管理することにより、個々のＣＰＵは均等な負荷状況となり、負荷分散が実現される。

【０００５】

【発明が解決しようとする課題】従来の並列計算機の負荷分散方法は以上のように行われているので、タスクを生成したＣＰＵと、タスクを実行するＣＰＵが互い違いになる場合がある。また並列計算機のＣＰＵはタスクの生成とタスクの割り当てを行う度に、共有のタスクキューにアクセスする必要がある。そのため通信路の通信負荷が大きく、並列計算機のＣＰＵの個数を増やしていくと、通信路の性能がボトルネックとなり、並列計算機の性能が発揮できないという問題点があった。

【０００６】この発明は上記のような問題点を解消する

ためになされたもので、通信路の通信負荷が小さく、スケーラビリティのある並列計算機の負荷分散方法を得ることを目的としており、さらに負荷分散のオーバーヘッドが少なく、ＣＰＵ構成の情報を局所化した並列計算機の負荷分散方法を得ることを目的とする。

【０００７】

【課題を解決するための手段】この発明の実施例１に係る並列計算機の負荷分散方法は、タスクの生成時に自分のＣＰＵで処理できない場合に、並列計算機内のＣＰＵの負荷状況を調べ、並列計算機内の負荷の小さいＣＰＵにタスクを転送し実行させるものである。

【０００８】また、この発明の実施例２に係る並列計算機の負荷分散方法は、タスクの生成時に自分のＣＰＵで処理できない場合に、ＣＰＵ間距離情報データベースからＣＰＵ間距離を取得し、並列計算機内のＣＰＵの負荷状況を調べ、並列計算機内の負荷が小さくＣＰＵ間距離の短いＣＰＵにタスクを転送し実行させるものである。

【０００９】また、この発明の実施例３に係る並列計算機の負荷分散方法は、あらかじめ何も処理を割り当てておかない予備ＣＰＵを備え、タスクの生成時に自分のＣＰＵで処理できない場合に、予備ＣＰＵにタスクを転送し実行させるものである。

【００１０】また、この発明の実施例４に係る並列計算機の負荷分散方法は、タスク配分ＣＰＵを備え、タスクの生成時に自分のＣＰＵで処理できない場合に、タスク配分ＣＰＵにタスクを転送し、タスク配分ＣＰＵが並列計算機内のＣＰＵの負荷状況を調べ、並列計算機内の負荷の小さいＣＰＵにタスクを転送し実行させるものである。

【００１１】また、この発明の実施例５に係る並列計算機の負荷分散方法は、タスク配分ＣＰＵを備え、タスクの生成時に自分のＣＰＵで処理できない場合に、タスク配分ＣＰＵにタスクを転送し、タスク配分ＣＰＵがＣＰＵ間距離情報データベースからＣＰＵ間距離を取得し、並列計算機内のＣＰＵの負荷状況を調べ、並列計算機内の負荷が小さくＣＰＵ間距離の短いＣＰＵにタスクを転送し実行させるものである。

【００１２】また、この発明の実施例６に係る並列計算機の負荷分散方法は、タスク配分ＣＰＵと、あらかじめ何も処理を割り当てておかない予備ＣＰＵを備え、タスクの生成時に自分のＣＰＵで処理できない場合に、タスク配分ＣＰＵにタスクを転送し、タスク配分ＣＰＵが予備ＣＰＵにタスクを転送し実行させるものである。

【００１３】

【作用】この発明の実施例１における並列計算機の負荷分散方法は、タスクの生成時に自分のＣＰＵで処理できない場合に、並列計算機内のＣＰＵの負荷状況を調べ、並列計算機内の負荷の小さいＣＰＵにタスクを転送し実行させることにより負荷分散を行う。

【００１４】また、この発明の実施例２における並列計算機の負荷分散方法は、タスクの生成時に自分のＣＰＵで処理できない場合に、ＣＰＵ間距離情報データベースからＣＰＵ間距離を取得し、並列計算機内のＣＰＵの負荷状況を調べ、並列計算機内の負荷が小さくＣＰＵ間距離の短いＣＰＵにタスクを転送し実行させることにより負荷分散を行う。

【００１５】また、この発明の実施例３における並列計算機の負荷分散方法は、あらかじめ何も処理を割り当てておかない予備ＣＰＵを備え、タスクの生成時に自分のＣＰＵで処理できない場合に、予備ＣＰＵにタスクを転送し実行させることにより負荷分散を行う。

【００１６】また、この発明の実施例４における並列計算機の負荷分散方法は、タスク配分ＣＰＵを備え、タスクの生成時に自分のＣＰＵで処理できない場合に、タスク配分ＣＰＵにタスクを転送し、タスク配分ＣＰＵが並列計算機内のＣＰＵの負荷状況を調べ、並列計算機内の負荷の小さいＣＰＵにタスクを転送し実行させることにより負荷分散を行う。

【００１７】また、この発明の実施例５における並列計算機の負荷分散方法は、タスク配分ＣＰＵを備え、タスクの生成時に自分のＣＰＵで処理できない場合に、タスク配分ＣＰＵにタスクを転送し、タスク配分ＣＰＵがＣＰＵ間距離情報データベースからＣＰＵ間距離を取得し、並列計算機内のＣＰＵの負荷状況を調べ、並列計算機内の負荷が小さくＣＰＵ間距離の短いＣＰＵにタスクを転送し実行させることにより負荷分散を行う。

【００１８】また、この発明の実施例６における並列計算機の負荷分散方法は、タスク配分ＣＰＵと、あらかじめ何も処理を割り当てておかない予備ＣＰＵを備え、タスクの生成時に自分のＣＰＵで処理できない場合に、タスク配分ＣＰＵにタスクを転送し、タスク配分ＣＰＵが予備ＣＰＵにタスクを転送し実行させることにより負荷分散を行う。

【００１９】

【実施例】

実施例１．図１は、この発明の実施例１による並列計算機の負荷分散方式のブロック図であり、図において、１ａ、１ｂ、１ｃはＣＰＵ、２ａ、２ｂ、２ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃに接続されているメモリ、３ａ、３ｂ、ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃが実行するタスクを保持しておくタスクキュー、４ａ、４ｂ、４ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃの負荷状況を表す負荷情報、５はメモリ２ａ、２ｂ、２ｃをＣＰＵ１ａ、１ｂ、１ｃで共有するための通信路である。

【００２０】次に動作について図２を参照しながら説明する。図２は図１のＣＰＵ１ａ、１ｂ、１ｃで動作するタスクの生成処理を示すフローチャートである。

【００２１】並列計算機の負荷分散を行うために、ＣＰＵ１ａ、１ｂ、１ｃは、タスクを生成する際に、図２に示すタスク生成処理を実行する。まずＣＰＵ１ａがタス

クを生成する場合について説明する。ＣＰＵ１ａは、Ｓ１ステップでタスクを生成した後、Ｓ２ステップで自分のＣＰＵの負荷状況を負荷情報４ａから読み取り、負荷状況が許容値以下か否かを判断する。負荷状況が許容値以下の場合は、生成されたタスクが自分のＣＰＵ１ａで処理可能であると見なして、Ｓ３ステップでタスクを自分のＣＰＵ１ａのタスクキュー３ａに登録し、タスク生成処理を終了する。負荷状況が許容値より大きい場合は、生成されたタスクが自分のＣＰＵ１ａで処理不可能であると見なして、Ｓ４ステップで並列計算機内の他のＣＰＵ１ｂ、１ｃの負荷状況を負荷情報４ｂ、４ｃから読み取り、Ｓ５ステップで読み取った負荷状況から負荷の一番小さいＣＰＵを選択する。ここでＣＰＵが選択されたとすると、Ｓ６ステップで選択したＣＰＵ１ｃのタスクキュー３ｃにタスクを登録し、タスク生成処理を終了する。ＣＰＵ１ｂ、１ｃがタスクを生成する場合も同様である。

【００２２】実施例２．図３は、この発明の実施例２による並列計算機の負荷分散方式のブロック図であり、図において、１ａ、１ｂ、１ｃはＣＰＵ、２ａ、２ｂ、２ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃに接続されているメモリ、３ａ、３ｂ、３ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃが実行するタスクを保持しておくタスクキュー、４ａ、４ｂ、４ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃの負荷状況を表す負荷情報、５はメモリ２ａ、２ｂ、２ｃをＣＰＵ１ａ、１ｂ、１ｃで共有するための通信路、６ａ、６ｂ、６ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃから別のＣＰＵまでの距離情報を含んだＣＰＵ間距離情報データベースである。

【００２３】次に動作について図４を参照しながら説明する。図４は図３のＣＰＵ１ａ、１ｂ、１ｃで動作するタスク生成処理を示すフローチャートである。

【００２４】並列計算機の負荷分散を行うために、ＣＰＵ１ａ、１ｂ、１ｃは、タスクを生成する際に、図４に示すタスク生成処理を実行する。まずＣＰＵ１ａがタスクを生成する場合について説明する。ＣＰＵ１ａは、Ｓ７ステップでタスクを生成した後、Ｓ８ステップで自分のＣＰＵの負荷状況を負荷情報４ａから読み取り、負荷状況が許容値以下か否かを判断する。負荷状況が許容値以下の場合は、生成されたタスクが自分のＣＰＵ１ａで処理可能であると見なして、Ｓ９ステップでタスクを自分のＣＰＵ１ａのタスクキュー３ａに登録し、タスク生成処理を終了する。負荷状況が許容値より大きい場合は、生成されたタスクが自分のＣＰＵ１ａで処理不可能であると見なして、Ｓ１０ステップで並列計算機内の他のＣＰＵ１ｂ、１ｃの負荷状況を負荷情報４ｂ、４ｃから読み取り、Ｓ１１ステップで負荷状況が許容値以下のＣＰＵがあるか否かを判断する。負荷状況が許容値以下のＣＰＵが無い場合は、Ｓ１２ステップで負荷の一番小さいＣＰＵを選択する。ここでＣＰＵ１ｃが選択されたとすると、Ｓ１３ステップで選択したＣＰＵ１ｃのタスクキュー３ｃにタスクを登録し、タスク生成処理を終了する。負荷状況が許容値以下のＣＰＵが有る場合は、Ｓ１４ステップでＣＰＵ間距離情報データベース６ａからＣＰＵ間距離を読み取り、負荷状況が許容値以下のＣＰＵのうちＣＰＵ間距離の一番短いＣＰＵを選択する。ここでＣＰＵ１ｂが選択されたとすると、Ｓ１５ステップで選択したＣＰＵ１ｂのタスクキュー３ｂにタスクを登録し、タスク生成処理を終了する。ＣＰＵ１ｂ、１ｃがタスクを生成する場合も同様である。

【００２５】実施例３．図５は、この発明の実施例３による並列計算機の負荷分散方式のブロック図であり、図において、１ａ、１ｂ、１ｃはＣＰＵ、２ａ、２ｂ、２ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃに接続されているメモリ、３ａ、３ｂ、３ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃが実行するタスクを保持しておくタスクキュー、４ａ、４ｂ、４ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃの負荷状況を表す負荷情報、５はメモリ２ａ、２ｂ、２ｃをＣＰＵ１ａ、１ｂ、１ｃで共有するための通信路、７は予備ＣＰＵ、８は予備ＣＰＵ７に接続されているメモリ、９は予備ＣＰＵ７が実行するタスクを保持しておくタスクキューである。

【００２６】次に動作について図６を参照しながら説明する。図６は図５のＣＰＵ１ａ、１ｂ、１ｃで動作するタスク生成処理を示すフローチャートである。

【００２７】並列計算機の負荷分散を行うために、ＣＰＵ１ａ、１ｂ、１ｃは、タスクを生成する際に、図６に示すタスク生成処理を実行する。まずＣＰＵ１ａがタスクを生成する場合について説明する。ＣＰＵ１ａは、Ｓ１６ステップでタスクを生成した後、Ｓ１７ステップで自分のＣＰＵの負荷状況を負荷情報４ａから読み取り、負荷状況が許容値以下か否かを判断する。負荷状況が許容値以下の場合は、生成されたタスクが自分のＣＰＵ１ａで処理可能であると見なして、Ｓ１８ステップでタスクを自分のＣＰＵ１ａのタスクキュー３ａに登録し、タスク生成処理を終了する。負荷状況が許容値より大きい場合は、生成されたタスクが自分のＣＰＵ１ａで処理不可能であると見なして、Ｓ１９ステップで予備ＣＰＵ７のタスクキュー９にタスクを登録し、タスク生成処理を終了する。ＣＰＵ１ｂ、１ｃがタスクを生成する場合も同様である。

【００２８】実施例４．図７は、この発明の実施例４による並列計算機の負荷分散方式のブロック図であり、図において、１ａ、１ｂ、１ｃはＣＰＵ、２ａ、２ｂ、２ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃに接続されているメモリ、３ａ、３ｂ、３ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃが実行するタスクを保持しておくタスクキュー、４ａ、４ｂ、４ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃの負荷状況を表す負荷情報、５はメモリ２ａ、２ｂ、２ｃをＣＰＵ１ａ、１ｂ、１ｃで共有するための通信

路、１０はタスク配分ＣＰＵ、１１はタスク配分ＣＰＵ１０に接続されているメモリである。

【００２９】次に動作について図８（ａ）、（ｂ）を参照しながら説明する。図８（ａ）、（ｂ）は負荷分散方式を示すフローチャートであり、図８（ａ）は図７のＣＰＵ１ａ、１ｂ、１ｃで動作するタスク生成処理を示し、図８（ｂ）は図７のタスク配分ＣＰＵ１０で動作するタスク配分処理を示す。

【００３０】並列計算機の負荷分散を行うために、ＣＰＵ１ａ、１ｂ、１ｃは、タスクを生成する際に、図８（ａ）に示すタスク生成処理を実行する。まずＣＰＵ１ａがタスクを生成する場合について説明する。ＣＰＵ１ａは、Ｓ２０ステップでタスクを生成した後、Ｓ２１ステップで自分のＣＰＵの負荷状況を負荷情報４ａから読み取り、負荷状況が許容値以下か否かを判断する。負荷状況が許容値以下の場合は、生成されたタスクが自分のＣＰＵ１ａで処理可能であると見なして、Ｓ２２ステップでタスクを自分のＣＰＵ１ａのタスクキュー３ａに登録し、タスク生成処理を終了する。負荷状況が許容値より大きい場合は、生成されたタスクが自分のＣＰＵ１ａで処理不可能であると見なして、Ｓ２３ステップでタスク配分ＣＰＵ１０にタスクを送信し、タスク生成処理を終了する。タスク配分ＣＰＵ１０は常に図８（ｂ）に示すタスク配分処理を実行しており、Ｓ２４ステップでＣＰＵ１ａが送信したタスクを受信する。そして、Ｓ２５ステップで並列計算機内のタスクを送信したＣＰＵ１ａ以外のＣＰＵ１ｂ、１ｃの負荷状況を負荷情報４ｂ、４ｃから読み取り、Ｓ２６ステップで読み取った負荷状況から負荷の一番小さいＣＰＵを選択する。ここでＣＰＵ１ｃが選択されたとすると、Ｓ２７ステップで選択したＣＰＵ１ｃのタスクキュー３ｃにタスクを登録する。ＣＰＵ１ｂ、１ｃがタスクを生成する場合も同様である。

【００３１】実施例５．図９は、この発明の実施例５による並列計算機の負荷分散方式のブロック図であり、図において、１ａ、１ｂ、１ｃはＣＰＵ、２ａ、２ｂ、２ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃに接続されているメモリ、３ａ、３ｂ、３ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃが実行するタスクを保持しておくタスクキュー、４ａ、４ｂ、４ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃの負荷状況を表す負荷情報、５はメモリ２ａ、２ｂ、２ｃをＣＰＵ１ａ、１ｂ、１ｃで共有するための通信路、６はＣＰＵ１ａ、１ｂ、１ｃ相互の距離情報を含んだＣＰＵ間距離情報データベース、１０はタスク配分ＣＰＵ、１１はタスク配分ＣＰＵ１０に接続されているメモリである。

【００３２】次に動作について図１０（ａ）、（ｂ）を参照しながら説明する。図１０（ａ）、（ｂ）は負荷分散方式を示すフローチャートであり、図１０（ａ）は図９のＣＰＵ１ａ、１ｂ、１ｃで動作するタスク生成処理を示し、図１０（ｂ）は図９のタスク配分ＣＰＵ１０で動作するタスク配分処理を示す。

【００３３】並列計算機の負荷分散を行うために、ＣＰＵ１ａ、１ｂ、１ｃは、タスクを生成する際に、図１０（ａ）に示すタスク生成処理を実行する。まずＣＰＵ１ａがタスクを生成する場合について説明する。ＣＰＵ１ａは、Ｓ２８ステップでタスクを生成した後、Ｓ２９ステップで自分のＣＰＵの負荷状況を負荷情報４ａから読み取り、負荷状況が許容値以下か否かを判断する。負荷状況が許容値以下の場合は、生成されたタスクが自分のＣＰＵ１ａで処理可能であると見なして、Ｓ３０ステップでタスクを自分のＣＰＵ１ａのタスクキュー３ａに登録し、タスク生成処理を終了する。負荷状況が許容値より大きい場合は、生成されたタスクが自分のＣＰＵ１ａで処理不可能であると見なして、Ｓ３１ステップでタスク配分ＣＰＵ１０にタスクを送信し、タスク生成処理を終了する。タスク配分ＣＰＵ１０は常に図１０（ｂ）に示すタスク配分処理を実行しており、Ｓ３２ステップでＣＰＵ１ａが送信したタスクを受信する。そして、Ｓ３３ステップで並列計算機内のタスクを送信したＣＰＵ１ａ以外のＣＰＵ１ｂ、１ｃの負荷状況を負荷情報４ｂ、４ｃから読み取り、Ｓ３４ステップで負荷状況が許容値以下のＣＰＵがあるか否かを判断する。負荷状況が許容値以下のＣＰＵが無い場合は、Ｓ３５ステップで負荷の一番小さいＣＰＵを選択する。ここでＣＰＵ１ｃが選択されたとすると、Ｓ３６ステップで選択したＣＰＵ１ｃのタスクキュー３ｃにタスクを登録する。負荷状況が許容値以下のＣＰＵが有る場合は、Ｓ３７ステップでＣＰＵ間距離情報データベース６からＣＰＵ間距離を読み取り、負荷状況が許容値以下のＣＰＵのうちＣＰＵ間距離の一番短いＣＰＵを選択する。ここでＣＰＵ１ｂが選択されたとすると、Ｓ３８ステップで選択したＣＰＵ１ｂのタスクキュー３ｂにタスクを登録する。ＣＰＵ１ｂ、１ｃがタスクを生成する場合も同様である。

【００３４】実施例６．図１１は、この発明の実施例６による並列計算機の負荷分散方式のブロック図であり、図において、１ａ、１ｂ、１ｃはＣＰＵ、２ａ、２ｂ、２ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃに接続されているメモリ、３ａ、３ｂ、３ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃが実行するタスクを保持しておくタスクキュー、４ａ、４ｂ、４ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃの負荷状況を表す負荷情報、５はメモリ２ａ、２ｂ、２ｃをＣＰＵ１ａ、１ｂ、１ｃで共有するための通信路、７は予備ＣＰＵ、８は予備ＣＰＵ７に接続されているメモリ、９は予備ＣＰＵが実行するタスクを保持しておくタスクキュー、１０はタスク配分ＣＰＵ、１１はタスク配分ＣＰＵ１０に接続されているメモリである。

【００３５】次に動作について図１２（ａ）、（ｂ）を参照しながら説明する。図１２（ａ）、（ｂ）は負荷分散方式を示すフローチャートであり、図１２（ａ）は図１１のＣＰＵ１ａ、１ｂ、１ｃで動作するタスク生成処

理を示し、図１２（ｂ）は図１１のタスク配分ＣＰＵ１０で動作するタスク配分処理を示す。

【００３６】並列計算機の負荷分散を行うために、ＣＰＵ１ａ、１ｂ、１ｃは、タスクを生成する際に、図１２（ａ）に示すタスク生成処理を実行する。まずＣＰＵ１ａがタスクを生成する場合について説明する。ＣＰＵ１ａは、Ｓ３９ステップでタスクを生成した後、Ｓ４０ステップで自分のＣＰＵの負荷状況を負荷情報４ａから読み取り、負荷状況が許容値以下か否かを判断する。負荷状況が許容値以下の場合は、生成されたタスクが自分のＣＰＵ１ａで処理可能であると見なして、Ｓ４１ステップでタスクを自分のＣＰＵ１ａのタスクキュー３ａに登録し、タスク生成処理を終了する。負荷状況が許容値より大きい場合は、生成されたタスクが自分のＣＰＵ１ａで処理不可能であると見なして、Ｓ４２ステップでタスク配分ＣＰＵ１０にタスクを送信し、タスク生成処理を終了する。タスク配分ＣＰＵ１０は常に図１２（ｂ）に示すタスク配分処理を実行しており、Ｓ４３ステップでＣＰＵ１ａが送信したタスクを受信する。そして、Ｓ４４ステップで予備ＣＰＵ７のタスクキュー９にタスクを登録する。ＣＰＵ１ｂ、１ｃがタスクを生成する場合も同様である。

【００３７】

【発明の効果】この発明の実施例１によれば、タスクの生成時に自分のＣＰＵで処理できない場合に、並列計算機内のＣＰＵの負荷状況を調べ、並列計算機内の負荷の小さいＣＰＵにタスクを転送し実行させることにより負荷分散を行うので、通信路へのアクセスは、タスクの生成時に自分のＣＰＵで処理できない場合にのみ行われるため、通信路の通信負荷が小さくなる。

【００３８】また、この発明の実施例２によれば、タスクの生成時に自分のＣＰＵで処理できない場合に、ＣＰＵ間距離情報データベースからＣＰＵ間距離を取得し、並列計算機内のＣＰＵの負荷状況を調べ、並列計算機内の負荷が小さくＣＰＵ間距離の短いＣＰＵにタスクを転送し実行させることにより負荷分散を行うので、通信路へのアクセスは、タスクの生成時に自分のＣＰＵで処理できない場合にのみ行われ、さらに通信路へのアクセスが近距離間で行われるため、通信路の通信負荷がさらに小さくなる。

【００３９】また、この発明の実施例３によれば、あらかじめ何も処理を割り当てておかない予備ＣＰＵを備え、タスクの生成時に自分のＣＰＵで処理できない場合に、予備ＣＰＵにタスクを転送し実行させることにより、負荷分散を行うので、通信路へのアクセスは、タスクの生成時に自分のＣＰＵで処理できない場合にのみ行われるため、通信路の通信負荷が小さくなる。さらにタスクを転送する先のＣＰＵがあらかじめ決められているため、負荷分散のオーバーヘッドが少なくなる。

【００４０】また、この発明の実施例４によれば、タスク配分ＣＰＵを備え、タスクの生成時に自分のＣＰＵで処理できない場合に、タスク配分ＣＰＵにタスクを転送し、タスク配分ＣＰＵが並列計算機内のＣＰＵの負荷状況を調べ、並列計算機内の負荷の小さいＣＰＵにタスクを転送し実行させることにより負荷分散を行うので、通信路へのアクセスは、タスクの生成時に自分のＣＰＵで処理できない場合にのみ行われるため、通信路の通信負荷が小さくなる。さらにＣＰＵがどのように構成されているかという情報はタスク配分ＣＰＵのみが把握していればよいため、ＣＰＵの増設による負荷分散方式の変更はタスク配分ＣＰＵの処理のみでよく、拡張性に優れる。

【００４１】また、この発明の実施例５によれば、タスク配分ＣＰＵを備え、タスクの生成時に自分のＣＰＵで処理できない場合に、タスク配分ＣＰＵにタスクを転送し、タスク配分ＣＰＵがＣＰＵ間距離情報データベースからＣＰＵ間距離を取得し、並列計算機内のＣＰＵの負荷状況を調べ、並列計算機内の負荷が小さくＣＰＵ間距離の短いＣＰＵにタスクを転送し実行させることにより負荷分散を行うので、通信路へのアクセスは、タスクの生成時に自分のＣＰＵで処理できない場合にのみ行われ、さらに通信路へのアクセスが近距離間で行われるため、通信路の通信負荷がさらに小さくなる。さらにＣＰＵがどのように構成されているかという情報はタスク配分ＣＰＵのみが把握していればよいため、ＣＰＵの増設による負荷分散方式の変更はタスク配分ＣＰＵの処理のみでよく、拡張性に優れる。

【００４２】また、この発明の実施例６によれば、タスク配分ＣＰＵと、あらかじめ何も処理を割り当てておかない予備ＣＰＵを備え、タスクの生成時に自分のＣＰＵで処理できない場合に、タスク配分ＣＰＵにタスクを転送し、タスク配分ＣＰＵが予備ＣＰＵにタスクを転送し実行させることにより負荷分散を行うので、通信路へのアクセスは、タスクの生成時に自分のＣＰＵで処理できない場合にのみ行われるため、さらにタスクを転送する先のＣＰＵがあらかじめ決められているため、負荷分散のオーバーヘッドが少なくなる。さらにＣＰＵがどのように構成されているかという情報はタスク配分ＣＰＵのみが把握していればよいため、ＣＰＵの増設による負荷分散方式の変更はタスク配分ＣＰＵの処理のみでよく、拡張性に優れる。

【図面の簡単な説明】

【図１】　この発明の実施例１を示すブロック図である。

【図２】　この発明の実施例１を示すフローチャートである。

【図３】　この発明の実施例２を示すブロック図である。

【図４】　この発明の実施例２を示すフローチャートである。

11　　12

【図５】　この発明の実施例３を示すブロック図である。

【図６】　この発明の実施例３を示すフローチャートである。

【図７】　この発明の実施例４を示すブロック図である。

【図８】　この発明の実施例４を示すフローチャートである。

【図９】　この発明の実施例５を示すブロック図である。

【図１０】　この発明の実施例５を示すフローチャートである。

【図１１】　この発明の実施例６を示すブロック図である。

【図１２】　この発明の実施例６を示すフローチャートである。

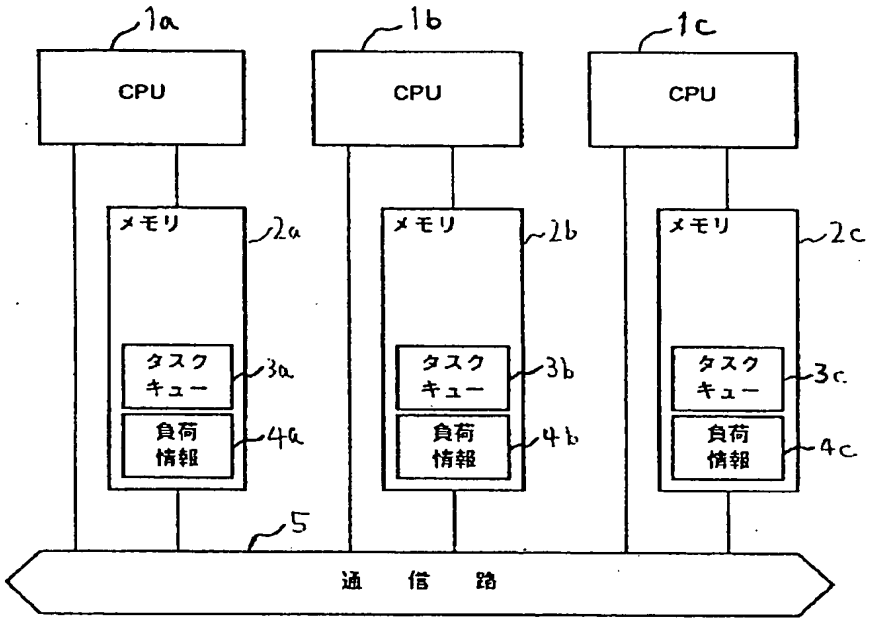
【図１３】　従来の並列計算機の負荷分散方法を示すブロック図である。

【図１４】　従来の並列計算機の負荷分散方法を示すフローチャートである。

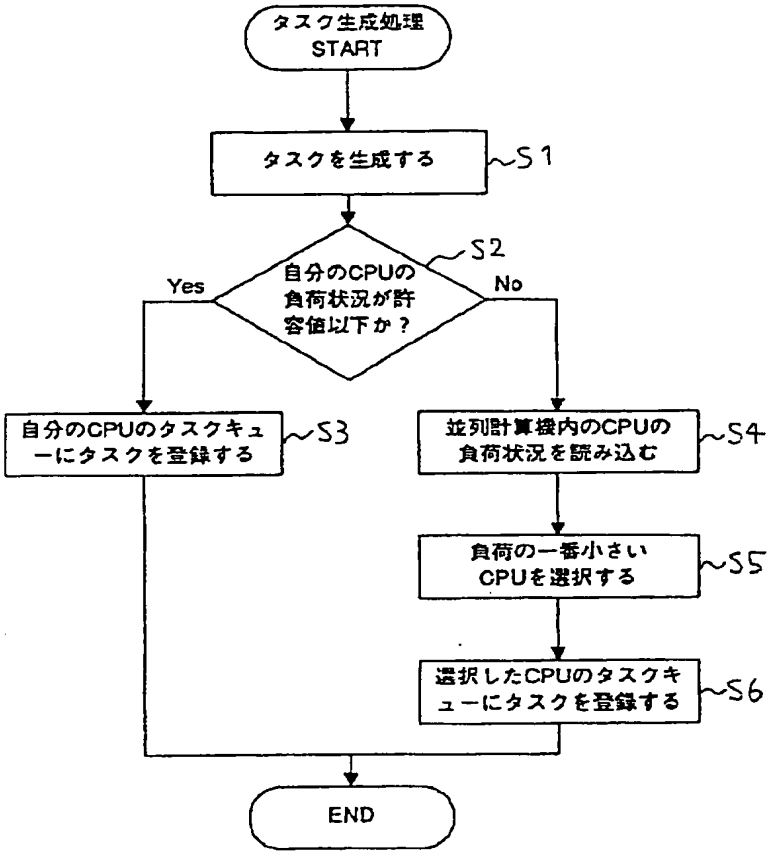
【符号の説明】

１　ＣＰＵ、２　メモリ、３　タスクキュー、４　負荷情報、５　通信路、６ＣＰＵ間距離情報データベース、７　予備ＣＰＵ、８　予備ＣＰＵのメモリ、９　予備ＣＰＵのタスクキュー、１０　タスク配分ＣＰＵ、１１　タスク配分ＣＰＵのメモリ。

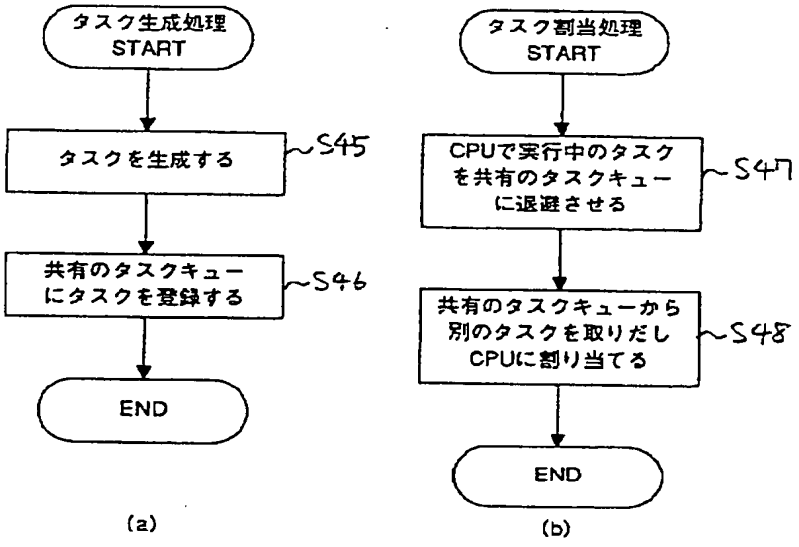
【図１】



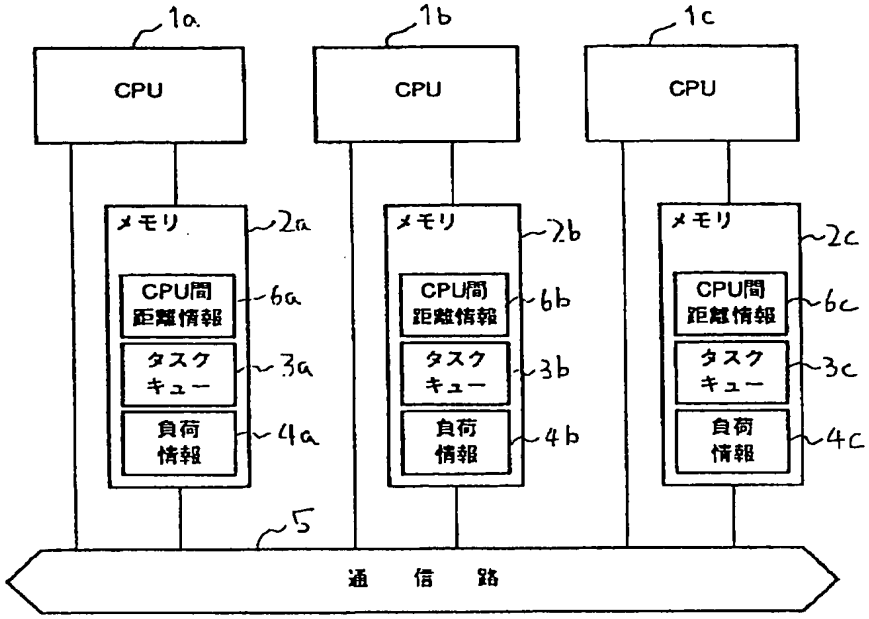
【図２】


タスク生成処理
START
タスクを生成する
S1
S2
自分のCPUの
負荷状況が許
容値以下か？
Yes
No
自分のCPUのタスクキュ
ーにタスクを登録する
S3
並列計算機内のCPUの
負荷状況を読み込む
S4
負荷の一番小さい
CPUを選択する
S5
選択したCPUのタスクキ
ューにタスクを登録する
S6
END

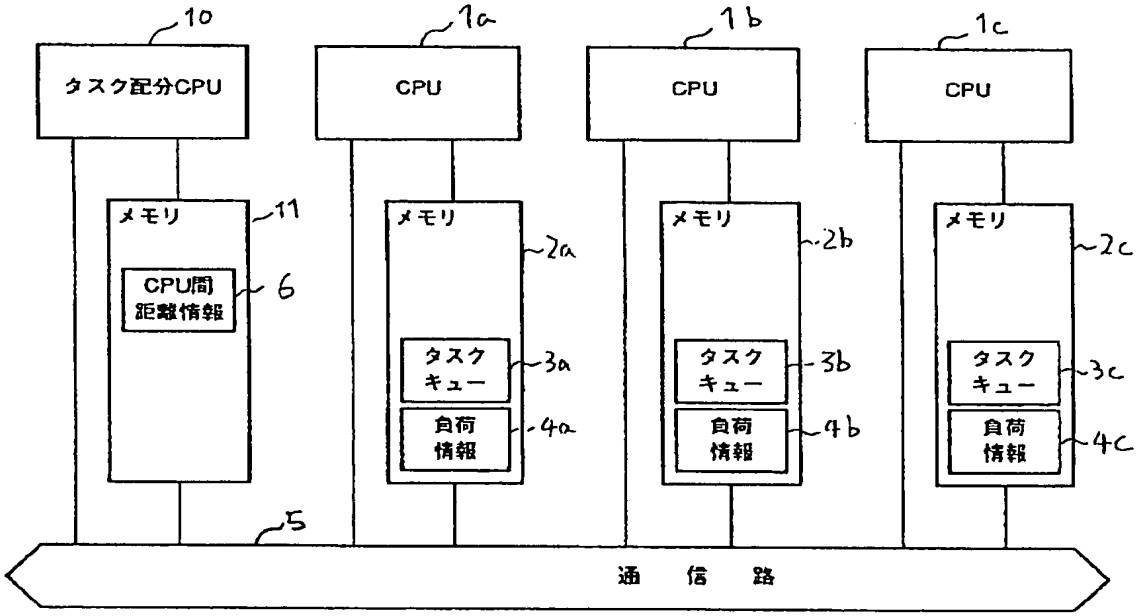
【図１４】


タスク生成処理
START
タスクを生成する
S45
共有のタスクキュー
にタスクを登録する
S46
END
(a)
タスク割当処理
START
CPUで実行中のタスク
を共有のタスクキュー
に退避させる
S47
共有のタスクキューから
別のタスクを取りだし
CPUに割り当てる
S48
END
(b)

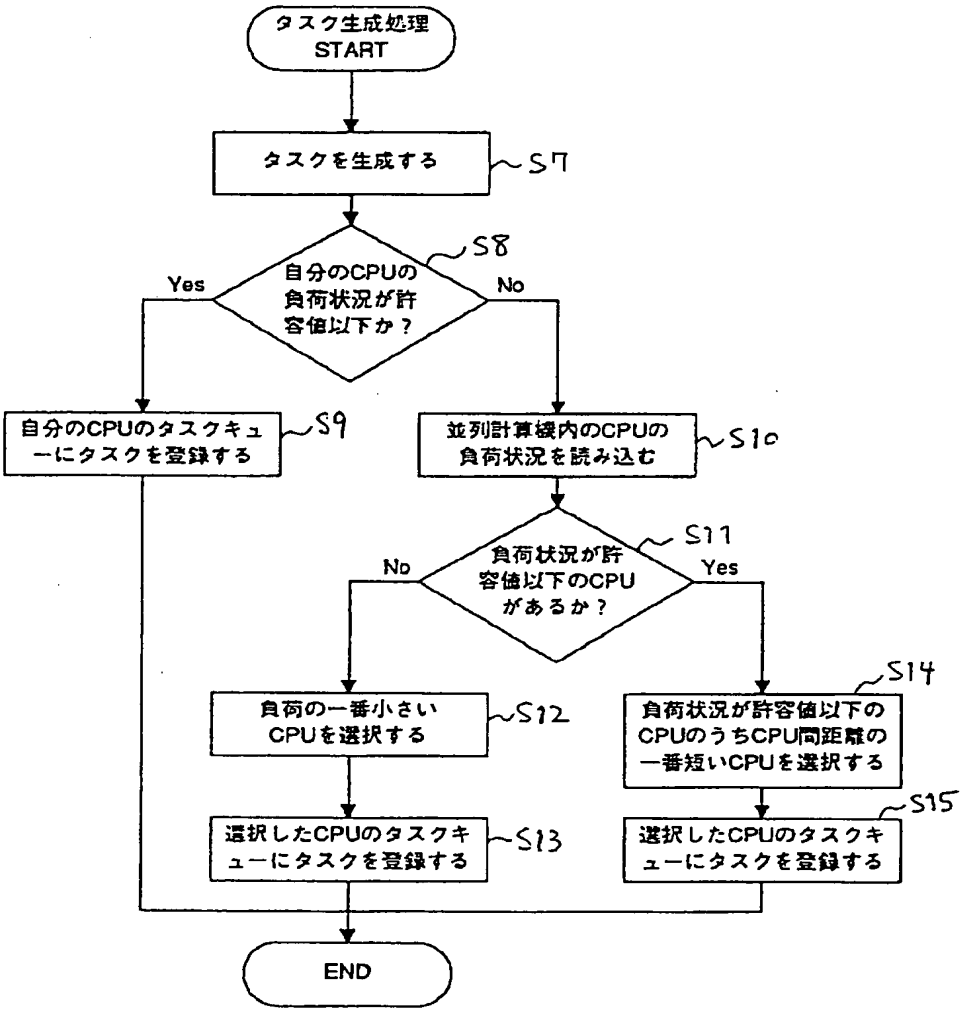
【図3】


1a
1b
1c
CPU
CPU
CPU
メモリ
メモリ
メモリ
2a
2b
2c
CPU間
距離情報
CPU間
距離情報
CPU間
距離情報
6a
6b
6c
タスク
キュー
タスク
キュー
タスク
キュー
3a
3b
3c
負荷
情報
負荷
情報
負荷
情報
4a
4b
4c
5
通　信　路

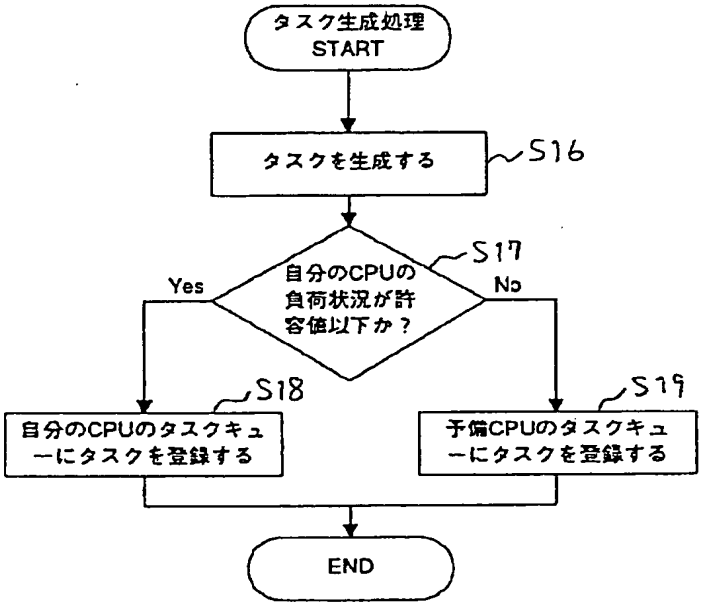
【図5】


10
1a
1b
1c
タスク配分CPU
CPU
CPU
CPU
メモリ
メモリ
メモリ
メモリ
11
2a
2b
2c
CPU間
距離情報
6
タスク
キュー
タスク
キュー
タスク
キュー
3a
3b
3c
負荷
情報
負荷
情報
負荷
情報
4a
4b
4c
5
通　信　路

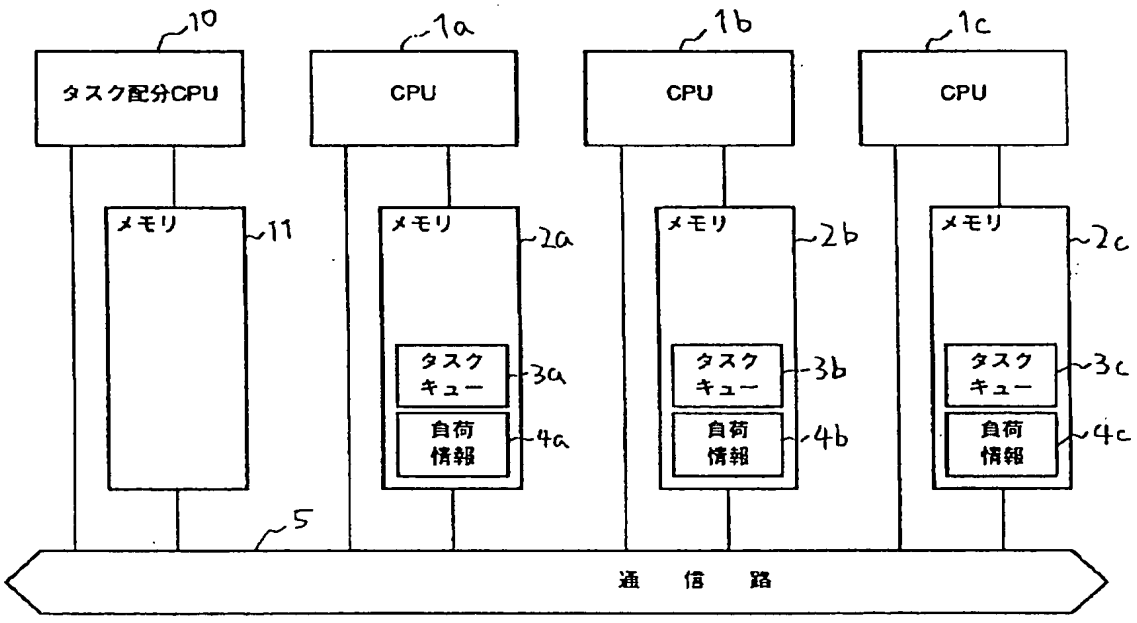
【図４】


タスク生成処理
START
タスクを生成する
S7
自分のCPUの
負荷状況が許
容値以下か？
S8
Yes
No
自分のCPUのタスクキュ
ーにタスクを登録する
S9
並列計算機内のCPUの
負荷状況を読み込む
S10
負荷状況が許
容値以下のCPU
があるか？
S11
No
Yes
負荷の一番小さい
CPUを選択する
S12
選択したCPUのタスクキ
ューにタスクを登録する
S13
負荷状況が許容値以下の
CPUのうちCPU間距離の
一番短いCPUを選択する
S14
選択したCPUのタスクキ
ューにタスクを登録する
S15
END

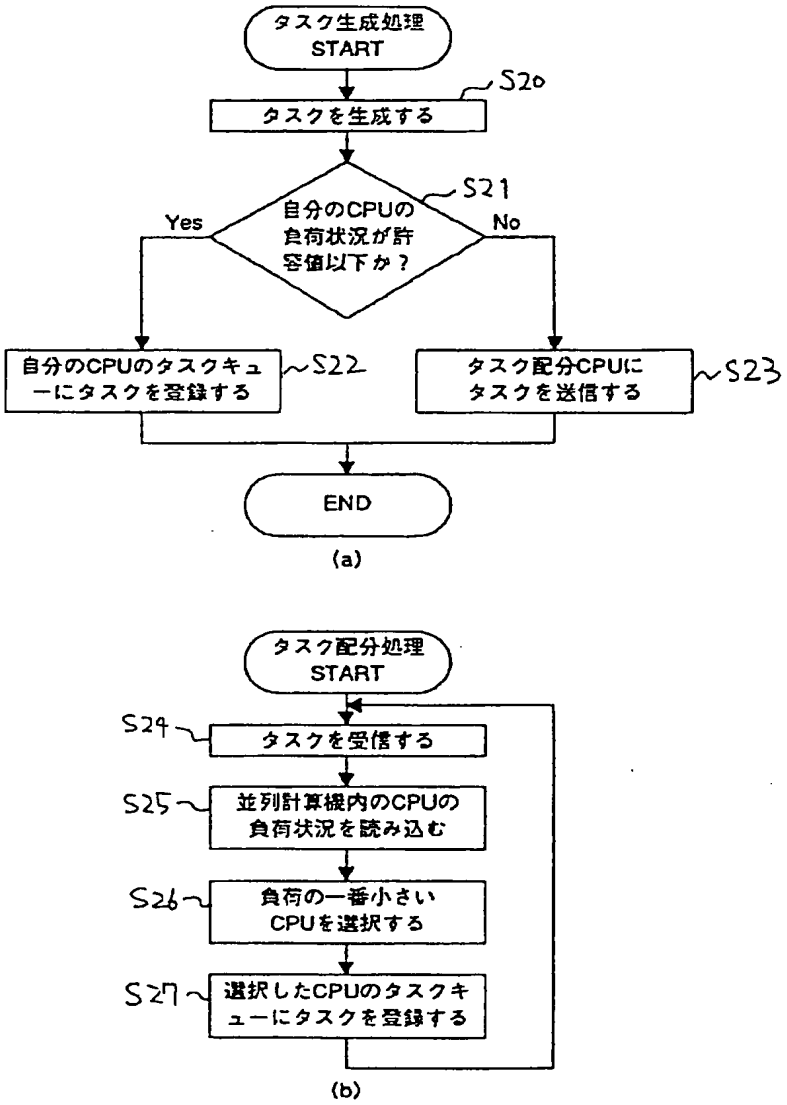
【図6】


タスク生成処理
START
タスクを生成する
S16
自分のCPUの
負荷状況が許
容値以下か？
S17
Yes
No
S18
自分のCPUのタスクキュ
ーにタスクを登録する
S19
予備CPUのタスクキュ
ーにタスクを登録する
END

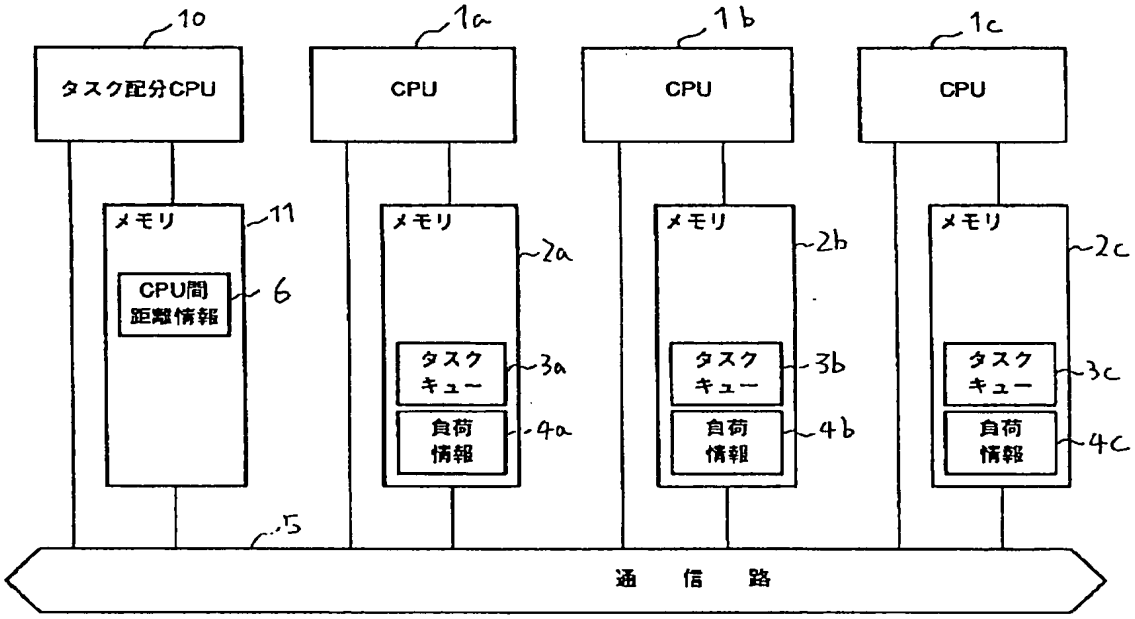
【図7】


10
タスク配分CPU
1a
CPU
1b
CPU
1c
CPU
メモリ
11
メモリ
2a
メモリ
2b
メモリ
2c
タスク
キュー
3a
タスク
キュー
3b
タスク
キュー
3c
負荷
情報
4a
負荷
情報
4b
負荷
情報
4c
5
通　信　路

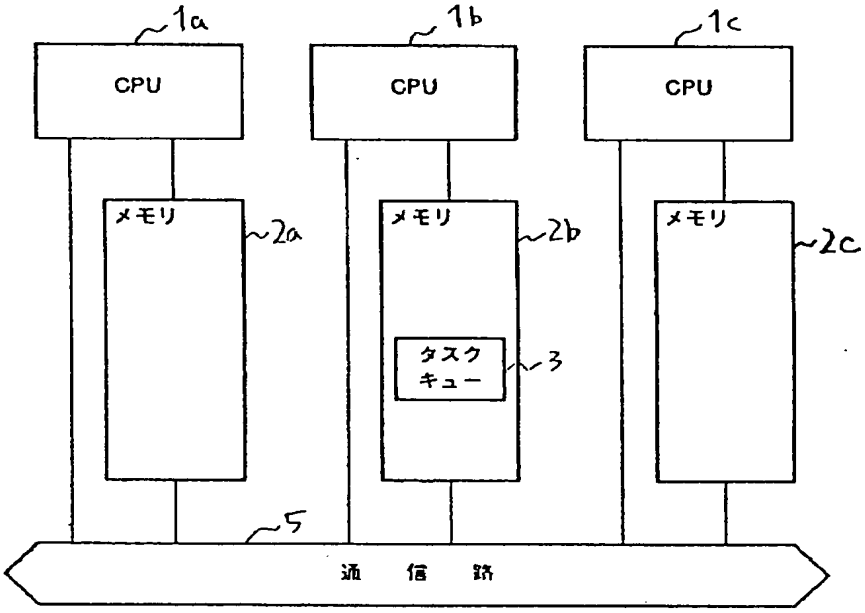
【図８】


タスク生成処理
START
S20
タスクを生成する
S21
自分のCPUの
負荷状況が許
容値以下か？
Yes
No
自分のCPUのタスクキュ
ーにタスクを登録する
S22
タスク配分CPUに
タスクを送信する
S23
END
(a)
タスク配分処理
START
S24
タスクを受信する
S25
並列計算機内のCPUの
負荷状況を読み込む
S26
負荷の一番小さい
CPUを選択する
S27
選択したCPUのタスクキ
ューにタスクを登録する
(b)

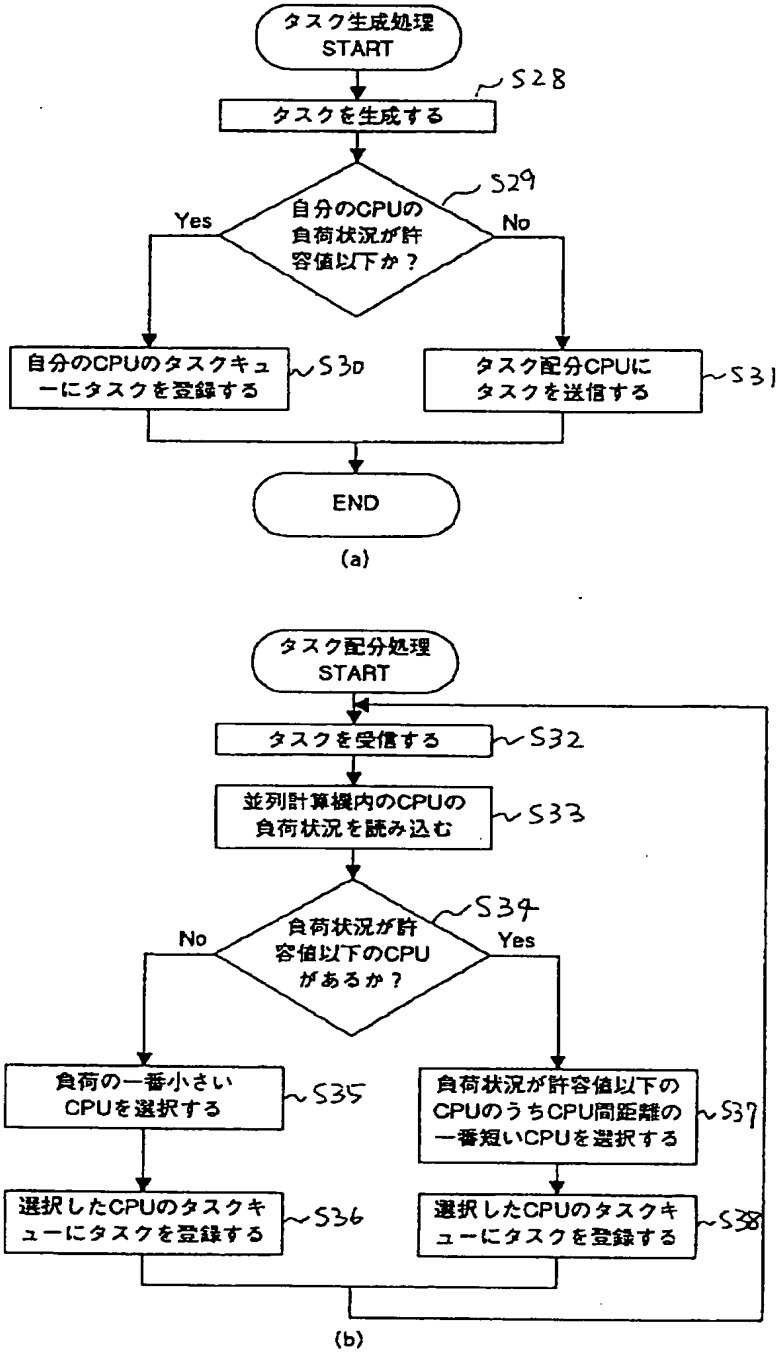
【図９】


10
タスク配分CPU
1a
CPU
1b
CPU
1c
CPU
メモリ
11
CPU間
距離情報
6
メモリ
2a
タスク
キュー
3a
負荷
情報
4a
メモリ
2b
タスク
キュー
3b
負荷
情報
4b
メモリ
2c
タスク
キュー
3c
負荷
情報
4c
5
通　信　路

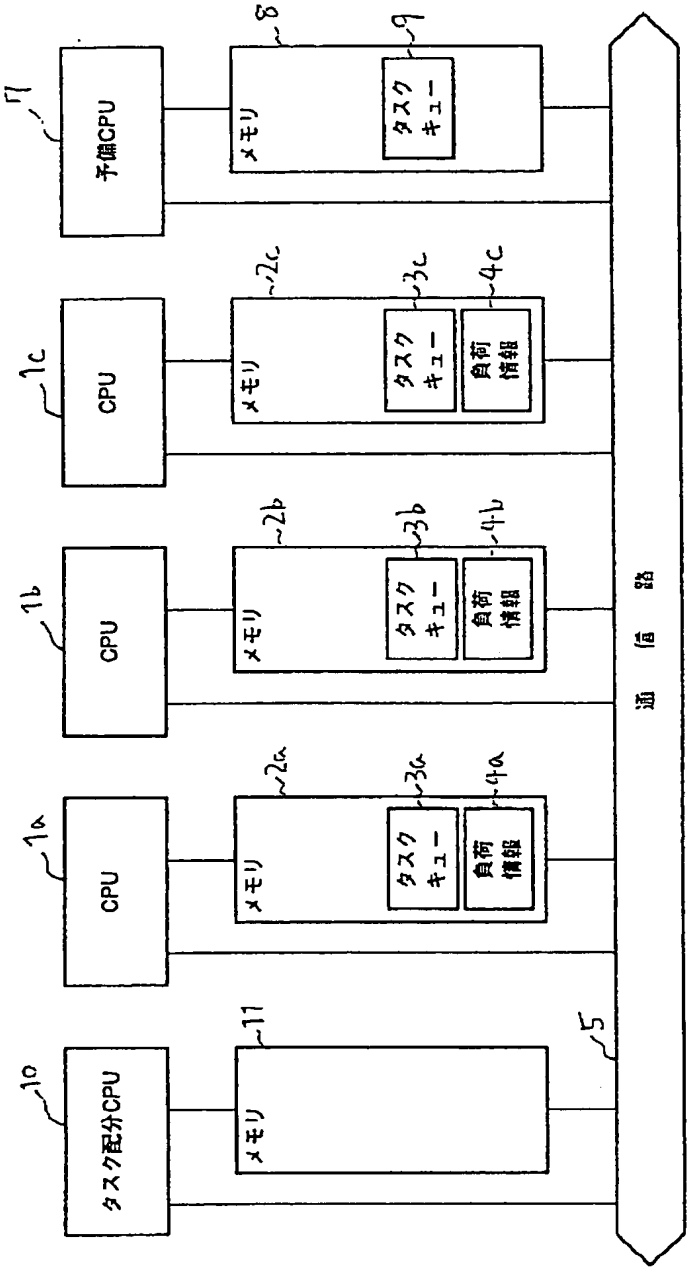
【図１３】


1a
CPU
1b
CPU
1c
CPU
メモリ
2a
メモリ
2b
タスク
キュー
3
メモリ
2c
5
通　信　路

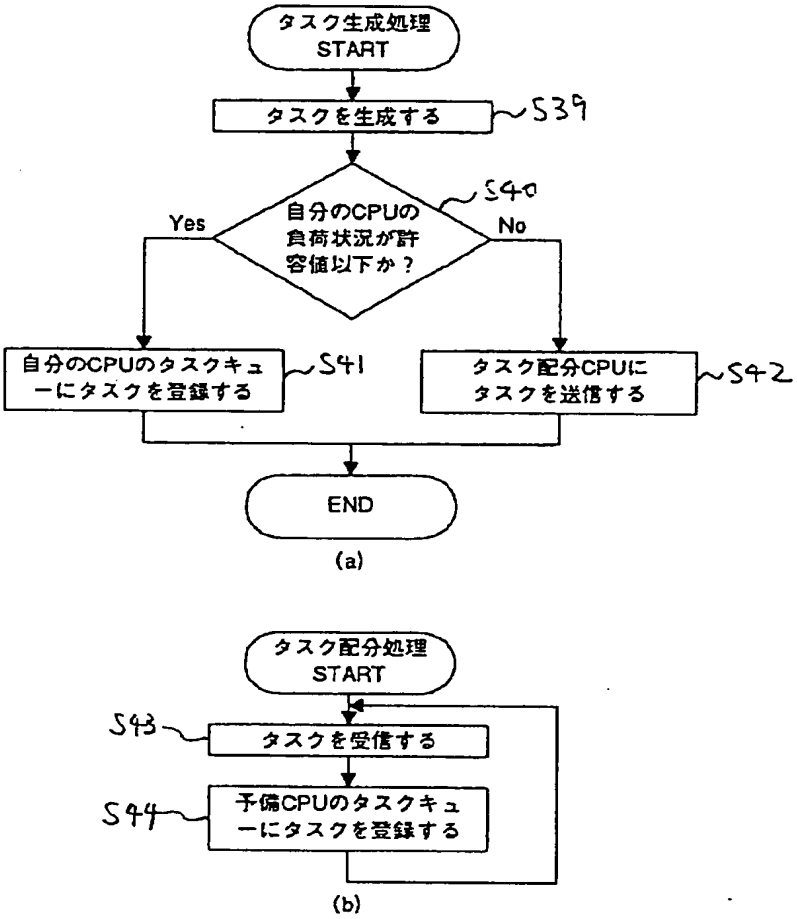
【図１０】


タスク生成処理
START
S28
タスクを生成する
S29
自分のCPUの
負荷状況が許
容値以下か？
Yes
No
自分のCPUのタスクキュ
ーにタスクを登録する
S30
タスク配分CPUに
タスクを送信する
S31
END
(a)
タスク配分処理
START
タスクを受信する
S32
並列計算機内のCPUの
負荷状況を読み込む
S33
S34
負荷状況が許
容値以下のCPU
があるか？
No
Yes
負荷の一番小さい
CPUを選択する
S35
選択したCPUのタスクキ
ューにタスクを登録する
S36
負荷状況が許容値以下の
CPUのうちCPU間距離の
一番短いCPUを選択する
S37
選択したCPUのタスクキ
ューにタスクを登録する
S38
(b)

【図１１】



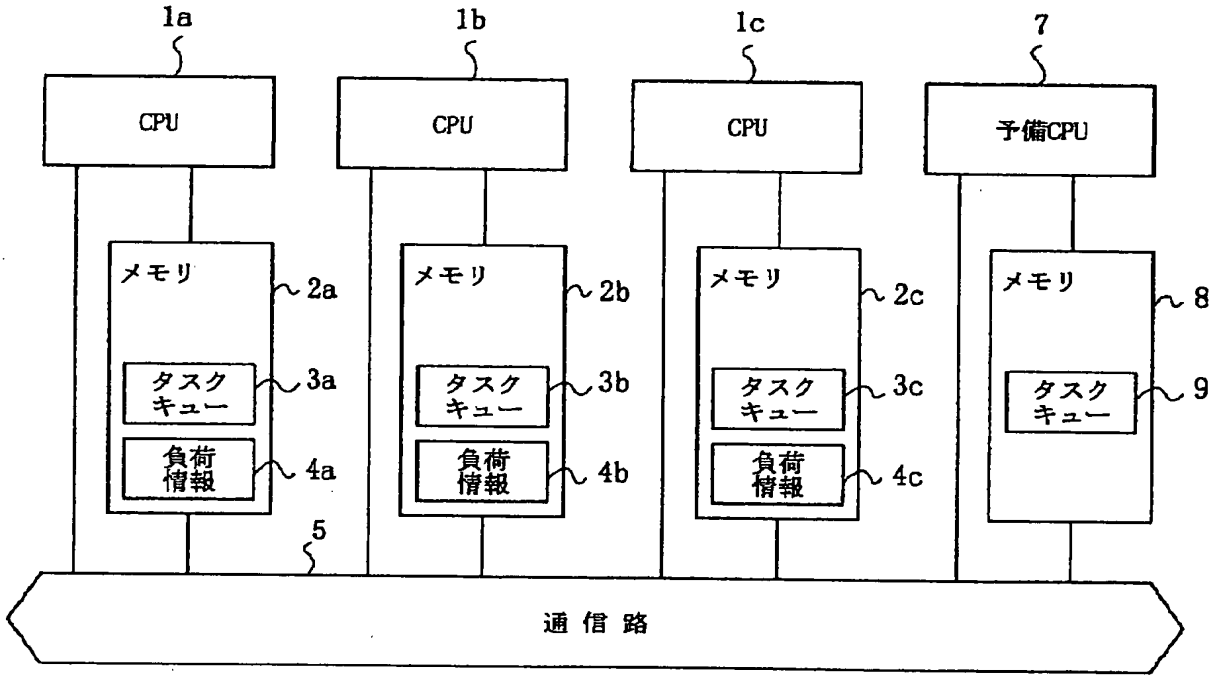
【図１２】


タスク生成処理
START
タスクを生成する
S39
自分のCPUの
負荷状況が許
容値以下か？
S40
Yes
No
自分のCPUのタスクキュ
ーにタスクを登録する
S41
タスク配分CPUに
タスクを送信する
S42
END
(a)
タスク配分処理
START
S43
タスクを受信する
S44
予備CPUのタスクキュ
ーにタスクを登録する
(b)

【公報種別】特許法第１７条の２の規定による補正の掲載
【部門区分】第６部門第３区分
【発行日】平成１３年６月２９日（２００１．６．２９）

【公開番号】特開平８－１６６９３１
【公開日】平成８年６月２５日（１９９６．６．２５）
【年通号数】公開特許公報８－１６７０
【出願番号】特願平６－３０８９０６
【国際特許分類第７版】
G06F 15/16 380
【ＦＩ】
G06F 15/16 380 Z

【手続補正書】
【提出日】平成１２年６月６日（２０００．６．６）
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】００１９
【補正方法】変更
【補正内容】
【００１９】
【実施例】実施例１．
図１は、この発明の実施例１による並列計算機の負荷分散方式のブロック図であり、図において、１ａ、１ｂ、１ｃはＣＰＵ、２ａ、２ｂ、２ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃに接続されているメモリ、３ａ、３ｂ、３ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃが実行するタスクを保持しておくタスクキュー、４ａ、４ｂ、４ｃはそれぞれＣＰＵ１ａ、１ｂ、１ｃの負荷状況を表す負荷情報、５はメモリ２ａ、２ｂ、２ｃをＣＰＵ１ａ、１ｂ、１ｃで共有するための通信路である。
【手続補正２】
【補正対象書類名】明細書
【補正対象項目名】００２１
【補正方法】変更
【補正内容】
【００２１】並列計算機の負荷分散を行うために、ＣＰＵ１ａ、１ｂ、１ｃは、タスクを生成する際に、図２に示すタスク生成処理を実行する。まずＣＰＵ１ａがタスクを生成する場合について説明する。ＣＰＵ１ａは、Ｓ１ステップでタスクを生成した後、Ｓ２ステップで自分のＣＰＵの負荷状況を負荷情報４ａから読み取り、負荷状況が許容値以下か否かを判断する。負荷状況が許容値以下の場合は、生成されたタスクが自分のＣＰＵ１ａで処理可能であると見なして、Ｓ３ステップでタスクを自分のＣＰＵ１ａのタスクキュー３ａに登録し、タスク生成処理を終了する。負荷状況が許容値より大きい場合は、生成されたタスクが自分のＣＰＵ１ａで処理不可能であると見なして、Ｓ４ステップで並列計算機内の他のＣＰＵ１ｂ、１ｃの負荷状況を負荷情報４ｂ、４ｃから読み取り、Ｓ５ステップで読み取った負荷状況から負荷の一番小さいＣＰＵを選択する。ここでＣＰＵ１ｃが選択されたとすると、Ｓ６ステップで選択したＣＰＵ１ｃのタスクキュー３ｃにタスクを登録し、タスク生成処理を終了する。ＣＰＵ１ｂ、１ｃがタスクを生成する場合も同様である。
【手続補正３】
【補正対象書類名】図面
【補正対象項目名】図５
【補正方法】変更
【補正内容】
【図５】


1a
1b
1c
7
CPU
CPU
CPU
予備CPU
メモリ
2a
メモリ
2b
メモリ
2c
メモリ
8
タスク
キュー
3a
タスク
キュー
3b
タスク
キュー
3c
タスク
キュー
9
負荷
情報
4a
負荷
情報
4b
負荷
情報
4c
5
通信路